NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AN ALGORITHM FOR GENERATING SHIP SCHEDULES
FOR A CRISIS DEPLOYMENT PROBLEM

by

Svein Buvik

September 1988

Thesis Advisor: Siriphong Lawphongpanich

Approved for public release; distribution is unlimited

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (If applicable) 55	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000
8a NAME OF FUNDING / SPONSORING ORGANIZATION 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
AN ALGORITHM FOR GENERATING SHIP SCHEDULES FOR A CRISIS DEPLOYMENT PROBLEM

12 PERSONAL AUTHOR(S)
Buvik, Svein

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM ____ TO ____	1988 September	47

16 SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Crisis Deployment, Deployment Plan, Schedule Generator, Dantzig-Wolfe Decomposition, Ship Schedule

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

A deployment is the movement of armed forces from their home bases to their strategic locations. The movement of these forces usually involves the transportation of military personnel as well as equipment and supplies. In a crisis situation, it is essential that the deployment is carried out in a expeditious manner.

This study considers the problem of constructing a deployment plan for sealift assets which transport military personnel, equipment, and supplies to their designated location in the least amount of time. In the construction of such a plan, feasible transportation schedules for each asset must be specified. When the number of movement requirements is large, the problem of arranging schedules for the assets is nontrivial. This thesis, therefore, describes an algorithm to generate these schedules. Based on several examples, this algorithm is shown to

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	
22a NAME OF RESPONSIBLE INDIVIDUAL Siriphong Lawphongpanich	22b TELEPHONE (Include Area Code) 408-646-2106 22c OFFICE SYMBOL 55Lp

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

☆ U.S. Government Printing Office 1986—606-24

19. Abstract (continued)

be effective and can be used in conjunction with algorithms for solving the overall deployment problem as well.

Approved for public release; distribution is unlimited.

An Algorithm For Generating Ship Schedules For A Crisis Deployment Problem

by

Svein Buvik
Lieutenant Commander, Royal Norwegian Navy
B.S., Norwegian Naval Academy, 1980

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 1988

ABSTRACT

A deployment is the movement of armed forces from their home bases to their strategic locations. The movement of these forces usually involves the transportation of military personnel as well as equipment and supplies. In a crisis situation, it is essential that the deployment is carried out in a expeditious manner.

This study considers the problem of constructing a deployment plan for sealift assets which transport military personnel, equipment, and supplies to their designated locations in the least amount of time. In the construction of such a plan, feasible transportation schedules for each asset must be specified. When the number of movement requirements is large, the problem of arranging schedules for the assets is nontrivial. This thesis, therefore, describes an algorithm to generate these schedules. Based on several examples, this algorithm is shown to be effective and can be used in conjunction with algorithms for solving the overall deployment problem as well.

TABLE OF CONTENTS

I. INTRODUCTION

A. PROBLEM STATEMENT

A deployment plan is a collection of schedules, one for each of the lift assets (trucks, trains, planes, and ships) assigned to the deployment. Each schedule normally comprises a list of cargoes arranged in the sequence which they are to be picked up from the ports of embarkation (POE's) and delivered to ports of debarkation (POD's) by the designated asset.

The generation of schedules becomes a great concern in the construction of optimal deployment plans because of the enormous number of possible schedules. Even for a deployment which consists of only 10 lift assets and 20 shiploads of cargoes, the number of possible schedules exceeds 10 million. Thus, regardless of the type of algorithm employed for the overall problem, the ability to produce an optimal deployment plan in a timely manner greatly depends on the ability to produce a good set of feasible schedules efficiently.

B. BACKGROUND

In mathematical programming, the idea of generating columns for the constraint matrix was first introduced by Dantzig and Wolfe [Ref. 1]. Among the first to employ the

column (schedule) generation (or the Dantzig-Wolfe decomposition) technique in the area of deployment are Rao and Zionts [Ref. 2] and Appelgren [Ref. 3]. Rao and Zionts employ the technique to solve the problem of allocating ships to existing schedules, and Applegren applies it to solve a ship scheduling problem.

More recently, Brown, Graves, and Ronen [Ref. 4] and Brown, Goodman, and Wood [Ref. 5] use a slightly different approach in solving the problem of scheduling oil tankers and naval surface combatants. Instead of generating the schedules as needed in the manner of the Dantzig-Wolfe decomposition, they have to generate all feasible schedules apriori because the schedules must satisfy many rules, restrictions, and relationships among the ports, ships, and cargoes. However, these rules, restrictions, and relationships also reduce the number of acceptable schedules to a manageable amount, thereby making the generation of schedules feasible.

C. OBJECTIVE

Most authors (see, Ronen [Ref. 6], Ronen [Ref. 7], Brown, Graves, and Ronen [Ref. 4], Brown, Goodman, and Wood [Ref. 5], Collier [Ref. 8], and Lally [Ref. 9]) consider a deployment with the minimum cost to be optimal. Instead of cost, one can also minimize other cost related quantities such as total distance traveled or the number of lift assets required. These quantities, if necessary, can always

be converted into a monetary amount. However, this study adopts a different optimality criterion which arises in the military deployment planning during the period of conflict (crisis). In this situation, the speed at which the armed forces, equipment, and supplies are deployed to the area of conflict is one of the most important factors governing how the conflict is resolved in the end. Therefore, it is essential that the plan can deliver all cargoes to their destinations in the shortest amount of time.

The objective of this study is to develop an efficient algorithm to generate schedules that can be used in conjunction with any column generation scheme for solving the crisis deployment problem. One such scheme is presented in a related Naval Postgraduate School Masters Thesis by Lt. N. R. Lima [Ref. 10].

The remainder of this thesis is divided into 4 chapters. Chapter 2 gives a summary of the mathematical formulation of the deployment planning problem. Chapter 3 describes the schedule generator. Chapter 4 discusses the computer implementation and the results from the experimentation with a group medium size problems. Finally, Chapter 5 summarizes the study and indicates areas for future research.

II. PROBLEM FORMULATION

In the deployment problem addressed below, all cargoes are assumed to be in full shiploads. This implies that the asset assigned to pick up a given cargo must deliver it before any other cargo can be picked up. Because of this assumption, the cargoes are also referred to as "movement requirements". Moreover, it is also assumed that other data essential to the problem, such as the distances between ports, ships' speeds, ships' initial positions, and tables indicating the ship/port and ship/cargo compatibility, are also given. Then, the problem of constructing a deployment plan with the minimum duration of time can be stated as follows:

Indices :

i - movement requirements (cargoes), where i = 1,....,M and M is the number of movement requirements.

j - ships, where j = 1,.....,N and N is the number of available lift assets.

k - feasible ship schedules. As mentioned in Section C of Chapter 1, the generation of these schedules is the topic of this thesis.

Data :

S_{ijk} = 1 if and only if the kth feasible schedule for the jth ship includes the ith movement requirement. Thus, S_{jk} is an M-dimensional vector representing the kth feasible schedule for ship j. A schedule for ship j is considered feasible if ship j is compatible with the cargoes it must pick up and with the ports it must visit.

t_{jk} = the completion time of schedule S_{jk}.

Decision variable :

x_{jk} = 1 if and only if the kth schedule for ship j is selected for the deployment.

Problem P1 :

$$\text{Min } \{ \max_{j} (\sum_{k} t_{jk} x_{jk} : j = 1,\ldots,N) \}$$

Subject to :

$$\sum_{j=1}^{N} \sum_{k} S_{ijk} x_{jk} \geq 1, \text{ for } i = 1,\ldots,M \qquad (1)$$

$$\sum_{k} x_{jk} \leq 1, \text{ for } j = 1,\ldots,N \qquad (2)$$

$$x_{jk} = 0 \text{ or } 1. \qquad (3)$$

In the above formulation constraint (1) forces each movement requirement to be picked up by at least one ship, while constraint (2) allows each ship to sail at most one

schedule. A set of x's satisfying constraints (1), (2), and (3), if it exists, would specify a deployment plan to support all the requirements of the deployment using the given assets.

In a related thesis, Lima [Ref. 10] obtains a solution to Problem P1 by parametrically solving a simpler problem several times. This simpler problem, Problem P2(τ) below, is called the feasibility-seeking problem and is a variant of Problem P1. In the feasibility-seeking problem, one assumes that a maximum time for deployment, τ, is given. Then, the task is to find a set of schedules with completion time less than or equal to τ that satisfies the constraints on Problem P1. To state the problem mathematically, let

Auxiliary Variables :

w_i = 1 if and only if movement requirement i is not delivered, and

Auxiliary Data :

$\underline{K}_j(\tau)$ = the index set of feasible schedules for ship j with the completion time less than or equal to τ, i.e.,

$\underline{K}_j(\tau) = \{ k : S_{jk}$ is a schedule and $t_{jk} \leq \tau \}$.

Then, the feasibility-seeking problem can be stated as follows:

Problem P2(τ) :

$$\text{Min} \quad \sum_i w_i$$

subject to :

$$\sum_{j=1}^{N} \sum_{k \epsilon \underline{K}_j(\tau)} S_{ijk} x_{jk} + w_i \geq 1, \text{ for } i = 1,\ldots,M \qquad (4)$$

$$\sum_k x_{jk} \leq 1, \text{ for } j = 1,\ldots,N \qquad (5)$$

$$x_{jk} = 0 \text{ or } 1. \qquad (6)$$

Associated with each constraint in (4) there is a dual variable u_i (≤ 0) which is referred to as the "cargo duals" in the subsequent chapters. Similarly, associated with each constraint in (5) there is a dual variable v_j (≥ 0) which will be referred to as the "ship dual".

Observe that for a given value τ, if the optimal objective function value of P2(τ) is zero, then the upper bound for Problem P1 is τ. On the other hand, if the optimal objective function value of P2(τ) is positive, then there is at least one undelivered shipload of cargo. Therefore, τ must be increased in order to obtain a feasible plan with the same number of assets. By varying τ and resolving Problem P2(τ) in a systematic manner, one can obtain a solution to P1.

Although $K_j(\tau)$ is a subset of the feasible schedules, it is still too large to enumerate. However, one can avoid this enumeration of schedules in the set $K_j(\tau)$ by applying the Dantzig-Wolfe decomposition technique to Problem P2(τ)

which produces a master and a subproblem. The master problem has the same form as Problem P2(τ) and the subproblem generates schedules (or columns) which belong to the set $K_j(\tau)$ and have negative reduced costs.

The following chapter describes an efficient algorithm for the subproblem. The details of solution techniques and strategies for the master problem are presented in the related thesis by Lima [Ref. 10].

III. A SCHEDULE GENERATOR

A. THE SUBPROBLEM

The subproblem resulting from the application of the Dantzig-Wolfe decomposition to Problem P2(τ) can be stated as follows:

Problem S1(j) :

$$k(j) = \arg \min_{k} \{ v_j + \sum_{i} S_{ijk} u_i : t_{jk} \leq \tau \}$$

As previously defined in Chapter 2, u_i and v_j are the cargo and ship duals of Problem P2(τ). For a given deployment plan, a negative cargo dual, u_i, indicates that the ith shipload is undelivered, and its magnitude can be regarded as the profit obtained from delivering the cargo to its destination. The ship dual, V_j, can be interpreted as the cost of using ship j. Therefore, the problem S1(j) would find a new schedule which utilizes ship j in the most profitable manner.

Moreover, Problem S1 is indexed by j to indicate that the generated schedule is for ship j. In theory one can solve one subproblem for a particular ship j, or solve N subproblems, one for each ship. To insure convergence only one schedule (column) with a negative reduced cost needs to be added to the master problem during each cycle of the decomposition process.

If the set of schedules S_{jk} is available apriori, solving Problem S1(j) is merely a matter of selecting one schedule from the set $K_j(\tau)$. Otherwise, Problem S1(j) resembles the vehicle routing problem [Ref. 11] with the exception that,

(1) the ship schedules do not have to start and terminate at the same port,

(2) the shiploads of cargoes must be delivered before any other shiploads can be picked up, and

(3) the schedule must be completed before time τ.

These three additional conditions greatly increase the complexity of Problem S1(j).

As stated, Problem S1(j) would always produce a schedule with the most negative reduced cost for each ship j. Moreover, considering the complexity of the problem, it may not be advantageous to solve Problem S1(j) to optimality. Moreover, the convergence of Dantzig-Wolfe decomposition only requires that the reduced cost of the new schedule (column) be negative. Thus, one only needs to search for a schedule, S_{jk}, which satisfies the following two conditions:

$$v_j + \sum_i S_{ijk} u_i < 0 \qquad (7)$$

$$t_{jk} \leq \tau \qquad (8)$$

where condition (7) ensures that the schedule has a

negative reduced cost and (8) guarantees that the schedule can be completed within the allowable time, τ.

In practice, entering columns which merely satisfy conditions (7) and (8) does not generally lead to an efficient algorithm. In the early iterations of the Dantzig-Wolfe algorithm, the ship duals, v_j, tend to be zero and the cargo duals are mostly negative. Thus, a schedule which picks up only one cargo with a negative dual would easily satisfy both conditions (7) and (8). However, such a schedule would be considered a bad schedule if the ratio of cargoes (movement requirements) to the number of ships is much larger than 1. For example, if the ratio is four then on the average one would expect each ship to complete four movement requirements. So, a schedule that picks up only one cargo is unlikely to be included in the optimal plan. The algorithm described below will search for one or more schedules which pick up a minimum number of cargoes as well as satisfying conditions (7) and (8). This minimum number of cargoes is calculated based on the ratio of cargoes to ships.

B. AN ALGORITHM WITH THE DEPTH-FIRST SEARCH STRATEGY

A schedule which satisfies conditions (7) and (8) and, at the same time, picks up the minimum number of cargoes is considered to be a "good" schedule. The algorithm below searches for one or more good schedules in the same manner as the branch and bound algorithm, with the depth-first

strategy, searches for an optimal solution for an integer program.

Basically, the algorithm tries to generate a sequence of numbers which is called a path. With the exception of the number zero, which represents the ship's initial position, the numbers in the path represent the cargo number and the order of the numbers represents the sequence in which the cargoes are to be picked up and delivered. Thus, a schedule or column for the master problem can easily be constructed from this path.

To illustrate the principle underlying the algorithm, consider a deployment problem with 3 movement requirements numbered 1 to 3. Figure 3-1 depicts a tree which represents all possible schedules for a ship. The node number in the tree corresponds to the movement requirement number except for the root node, which is numbered zero and corresponds to the ship's initial position. By traversing this tree starting at the root node, one would trace out a path from which a schedule can be constructed. For example, consider path 1 in Figure 3-1. This path corresponds to the following sequence of numbers: 0-1-3-2 which in turn corresponds to a schedule for the ship to complete movement requirements 1,3, and 2 in sequence. Similarly, path 2 corresponds to a schedule for the ship to complete movement requirements 2 and 3 in sequence.

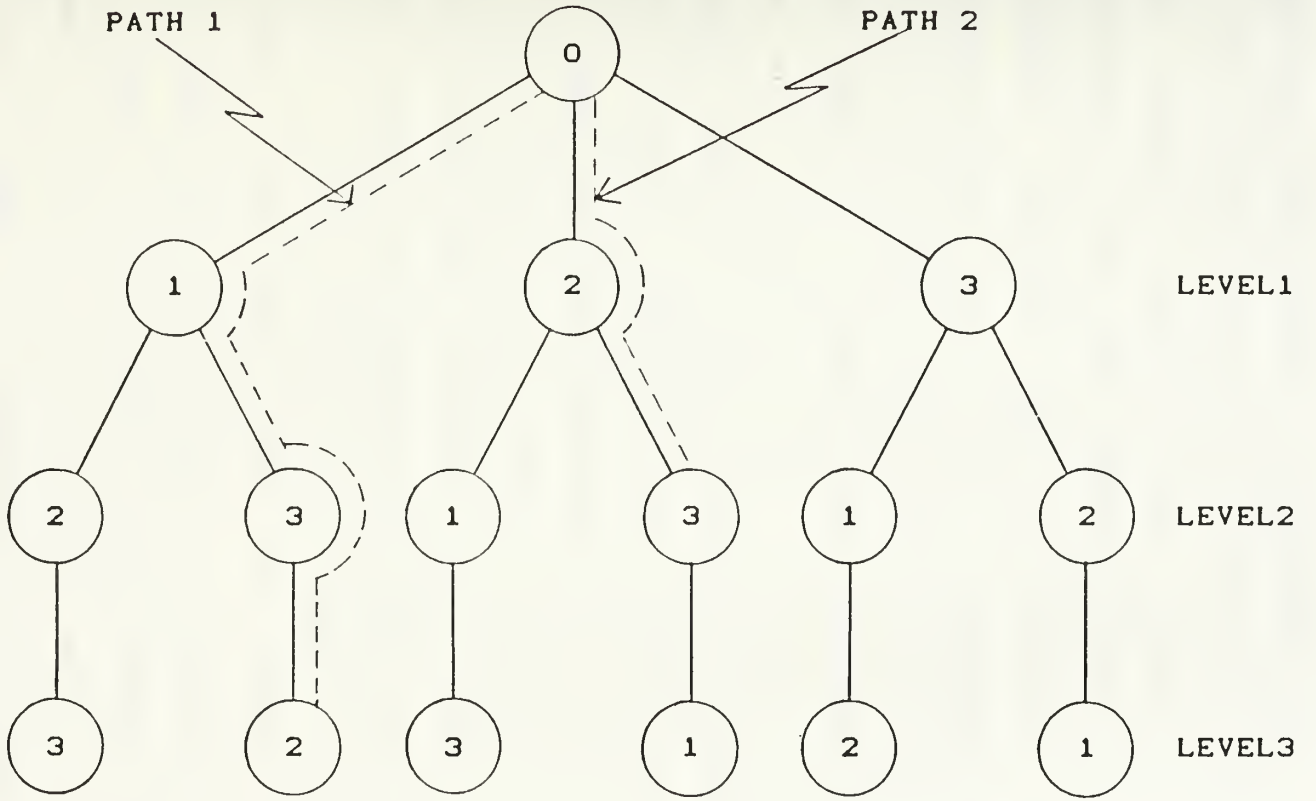

FIGURE 3-1

A tree representing all possible schedules with 3 movement requirements.

Initially, the algorithm discards all movement requirements which are either incompatible with the ship in consideration or have a positive dual variable. This incompatibility is due to either ship/port or ship/cargo incompatibility. Assume that the remaining movement requirements are numbered as 4, 6, 7, and 9. Then, starting at node zero with the current path containing only the number 0, the algorithm creates nodes 4, 6, 7, and 9 and attaches them to node zero (see Figure 3-2). From node zero, the algorithm considers adding a new movement requirement to the current path, say node 4; this is called branching. At node 4, the algorithm checks to see if the schedule which contains movement requirement 4 can be completed in τ days. If the answer is yes, the algorithm then includes node 4 into the current path, creates nodes corresponding to movement requirements which are not members of the current path, and attaches them to node 4. These nodes are node 6, 7, and 9. At this point, the algorithm would branch to, say, node 6 and at node 6 it checks to see if the schedule which contains movement requirements 4 and 6 can be completed in τ days. If no, the node is not added to the current path. The algorithm would then check if the current path, 0-4, corresponds to a schedule with negative reduced cost and picks up the minimum number of shiploads of cargo. If no, node 6 is

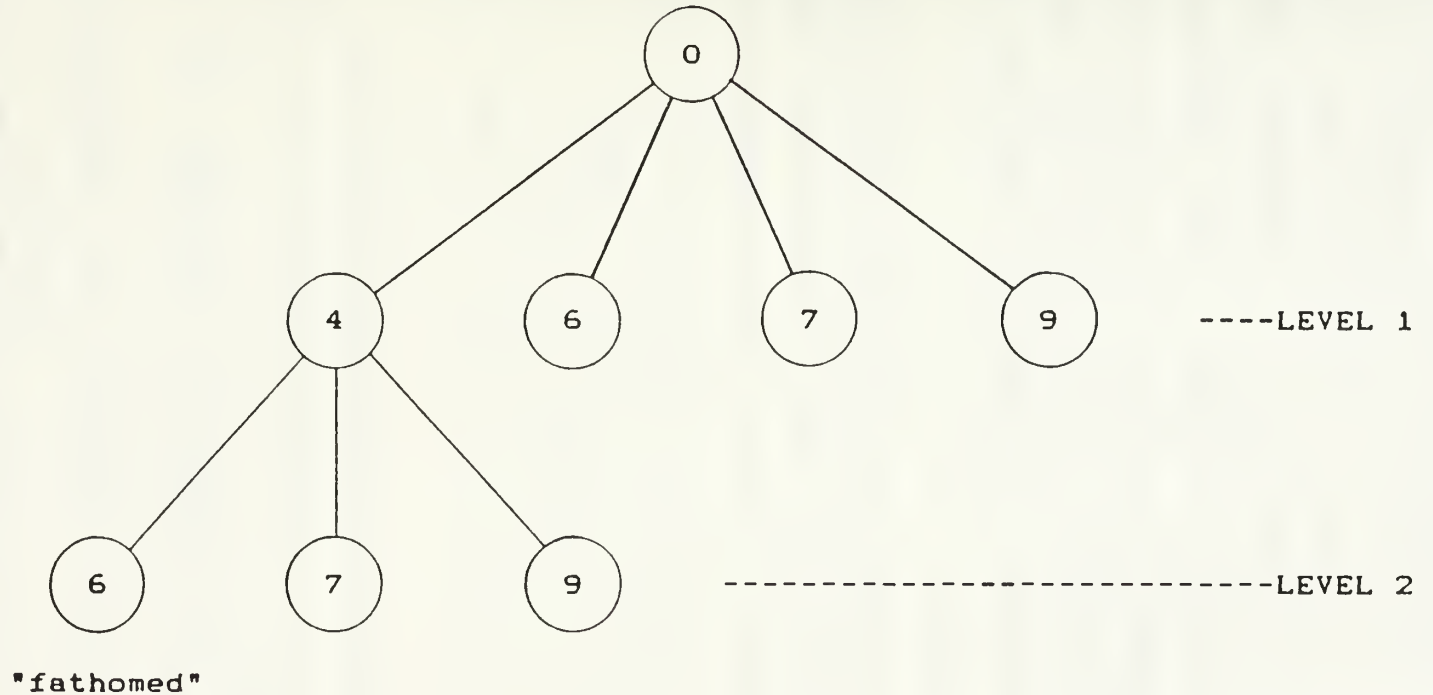

FIGURE 3-2

A tree to illustrate the branching process of the algorithm.

"fathomed" and the algorithm would branch to other unfathomed nodes in the depth-first manner, which in this case would be node 7 or node 9 in the second level. If yes, then a good schedule is found, and the algorithm would stop if only one good schedule is desired. If more schedules are required, the algorithm would store this good schedule, fathom the node, and branch to other unfathomed nodes to obtain the desired number of schedules or until all nodes in the tree are fathomed.

Below, we formally state the algorithm discussed above. It employs the abstract data type called STACKS [Ref. 12] to dynamically generate the search tree and conserve memory requirements.

Algorithm

Input :

j =	the index of the ship being considered.
τ =	the maximum allowable completion time for any schedule.
M =	the total number of shiploads of cargo (movement requirements).
NCOL =	the required number of "good" schedules for ship j.
NPICK =	the minimum number of cargoes to be picked up by each good schedule.
v_j =	the dual price for ship j.
u_j =	the dual price for movement requirement i.

c_{ij} = 1 if movement requirement i is compatible with ship j

= 0 otherwise.

Step 1 : For each i = 1,...,M, set

$\bar{u}_i = u_i * c_{ij} + 99*(1 - c_{ij})$.

Remove all movement requirements i with $\bar{u}_i > 0$ from further consideration, and let M' be the remaining movement requirements, i. e., those with $\bar{u}_i \leq 0$.

Step 2 : Sort $\bar{u}_i$ in an ascending order.

Step 3 : Set KOUNT = 0, P = {0}, and initialize the STACK.

Step 4 : Put the movement requirements into the STACK in the same order as the sorted dual variables $\bar{u}_i$.

Step 5 : While the STACK is not empty, do the following:

5.1) Remove the movement requirement, say, k, from the top of the STACK.

5.2) Calculate the completion time for path P U {k}.

5.3) If this completion time s g, then do the following:

a) Set P = P U {k}.

b) Put the movement requirements not in the path on top of the STACK in the order of decreasing $\bar{u}_i$.

c) Go to Step 5.1

5.4) Otherwise, calculate the reduced cost for the schedule corresponding to path P.

a) If the reduced cost is negative and the number of movement requirements in P r NPICK, then a good schedule is found. Set KOUNT = KOUNT + 1.

If KOUNT s NCOL, remove node k from the STACK and go to Step 5.1.

Otherwise, return to the master problem.

b) If either the reduced cost is nonnegative or the number of movement requirements in P is less than NPICK, then remove node k from the STACK and go to Step 5.1.

Note that Step 5.3.b dynamically generates the new level of nodes in the search tree. Moreover, the order in which the nodes are put on top of the STACK allows the search to progress in the depth-first manner with the movement requirement with the most negative cargo dual being searched first. In Step 5.4, the removal of node k from the STACK is equivalent to fathoming the node in the search tree.

IV. COMPUTATIONAL RESULTS

The algorithm for generating schedules presented in Chapter 3 was implemented in VS FORTRAN on the IBM 3033AP computer at the W. R. Church Computing Center of the Naval Postgraduate School. The master problem is also implemented in the same manner and is fully described in the Master's thesis by Lima [Ref. 10].

For the computational experiments, we consider three problems which contain approximately the same number of lift assets and movement requirements as one would expect in a deployment of a small armed force such as that of Norway. Problem 1 has 48 movement requirements and 25 lift assets, Problem 2 has 48 movement requirements and 30 lift assets, and Problem 3 has 60 movement requirements and 30 lift assets. The ports of embarkation (New York, Norfolk, Charleston, Jacksonville, Pensacola) and debarkation (Hamburg, Wilhelmshaven, Rotterdam, Antwerpen, Cherebourg) for the movement requirements are given in Table 4-1. Table 4-2 and 4-3 give the distances between the various ports and the initial position of each asset. The speed for the assets are between 13 and 22 knots, and the probability that the ship is compatible with a given movement requirement is .75.

TABLE 4-1 : NUMBER OF MOVEMENT REQUIREMENTS BETWEEN POE'S AND POD'S.

		HAM.	WILH.	ROT.	ANTW.	CHB.
PROBLEM 1 and 2	N.Y.	2	2	2	2	2
	NORF.	3	2	2	2	2
	CHAR.	2	2	2	2	2
	JAX.	2	3	3	2	3
	PENS.	0	0	2	2	0

		HAM.	WILH.	ROT.	ANTW.	CHB.
PROBLEM 3	N.Y.	4	4	3	2	2
	NORF.	2	2	2	3	2
	CHAR.	3	3	2	2	2
	JAX.	3	2	2	2	3
	PENS.	0	1	3	4	2

TABLE 4-2 : DISTANCE BETWEEN POE'S AND POD'S IN NAUTICAL MILES.

	HAM.	WILH.	ROT.	ANTW.	CHB.
N.Y.	4030	3950	3790	3775	3520
NORF.	4340	4260	4090	4075	3800
CHAR.	4650	4560	4390	4370	4090
JAX.	4850	4770	4590	4570	4280
PENS.	5390	5300	5125	5110	4820

TABLE 4-3. INITIAL DISTANCE BETWEEN SHIPS AND PORTS

SHIP #	N.Y.	NORF.	CHAR.	JAX.	PENS.
1	0	245	550	720	1190
2	0	245	550	720	1190
3	300	545	850	1020	1490
4	300	300	500	700	1100
5	800	800	900	1000	1400
6	100	350	650	820	1290
7	245	0	300	475	975
8	245	0	300	475	975
9	200	200	400	600	1100
10	600	600	700	900	1400
11	150	100	400	575	1075
12	350	100	200	375	875
13	550	300	0	165	700
14	550	300	0	165	700
15	800	750	700	750	1100
16	450	200	100	265	900
17	350	250	300	350	750
18	1200	1100	1000	1100	1500
19	720	475	165	0	600
20	720	475	165	0	600
21	1100	1000	900	900	1200
22	920	675	365	200	400
23	450	350	300	400	800
24	1350	1250	1200	1200	1600
25	1190	975	700	600	0
26	1190	975	700	600	0
27	890	675	400	300	300
28	1290	1075	800	700	300
29	900	700	500	400	600
30	1090	875	600	500	100

A. A HEURISTIC FOR SELECTING MOVEMENT REQUIREMENTS

In Step 1, the algorithm discards those movement requirements which have a positive dual value because of incompatibility with the ship under consideration. The remaining movement requirements are then selected for branching based solely on the magnitude of the associated cargo duals. Intuitively, this seems inefficient. Assume that the algorithm is constructing a new schedule for ship j. Then those movement requirements which are in the schedule for ship j and those which are still unassigned at the end of the preceding master problem should be selected for branching first. Otherwise, movement requirements which have been assigned to other ships may be assigned to the new schedule for ship j also. This would create a "double coverage" for the same movement requirement. To create the least amount of double coverage a negative number is added to the cargo duals for the movement requirements which are already assigned to ship j and to those which have yet to be assigned to any ship. This negative number should be large enough to insure that when the movement requirements are ordered in Step 2 and put into the STACK in Step 3 and Step 5.3.b, the desired movement requirements are always above the others in the STACK.

To implement the above heuristic, Step 1 of the algorithm in Chapter 3 is modified as follows:

Step 1 :

a) For each i = 1,..., M set

$$\bar{u}_i = \begin{cases} 99 & \text{if } c_{ij} = 0. \\ u_i & \text{if } c_{ij} = 0 \text{ and cargo i has been assigned to another ship l, where l = j, by the previous master problem iteration.} \\ -\delta & \text{if } c_{ij} = 0 \text{ and cargo i is unassigned or has been assigned to ship j by the previous master problem iteration.} \end{cases}$$

where δ is a sufficiently large positive number.

b) Remove all movement requirements i with $\bar{u}_i > 0$ from further consideration, and let M' be the remaining movement requirements, i.e., those with $\bar{u}_i \leq 0$.

The algorithm in Chapter 3 has been implemented both with and without the above heuristic for selecting movement requirements. Figure 4-4 shows that, based on the average CPU time to solve the three sample problems with various values for τ, the algorithm with the heuristic clearly outperforms the one without. In fact, the algorithm with the heuristic is on the average 47% faster. Henceforth, the subproblem algorithm is implemented with the heuristic for selecting movement requirements.

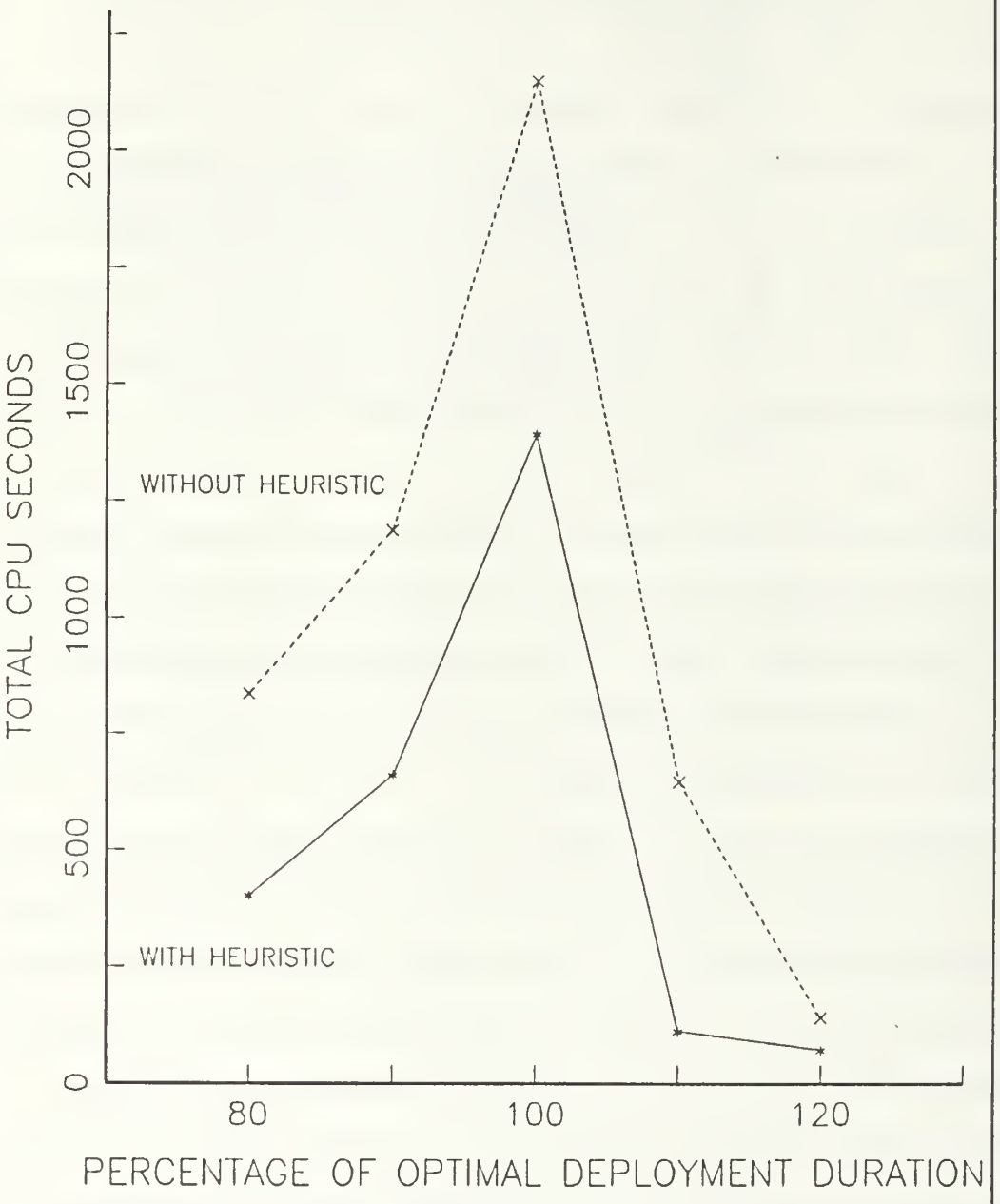

Figure 4-4

Comparison of total CPU time with and without heuristic implemented.

B. THE NUMBER OF SCHEDULES TO GENERATE FOR THE MASTER PROBLEM

In general, it is unclear what is the optimal number of columns to generate during each cycle of the decomposition process. More columns mean more information for the master problem during each cycle. At the same time, more columns in the master problem also mean more CPU time for pricing out and other housekeeping operations. However, Figure 4-5 demonstrates that it is advantageous to generate between 10 and 20 schedules (columns) for Problem 2 when τ = 29 days (the optimal duration).

Table 4-6 provides an explanation for the above finding. The table lists the average number of nodes in the search tree for the two strategies: one column and ten columns. Although the average number of nodes in the search tree for the ten column strategy is more than the one column strategy in the early iterations, the information provided by the additional nine columns actually helps the master problem to find a solution quicker. Moreover, as the number of iterations grow, the one column strategy occasionally generates unacceptably large search trees in order to find one negative reduced cost column which could have been generated earlier if the ten columns strategy was employed.

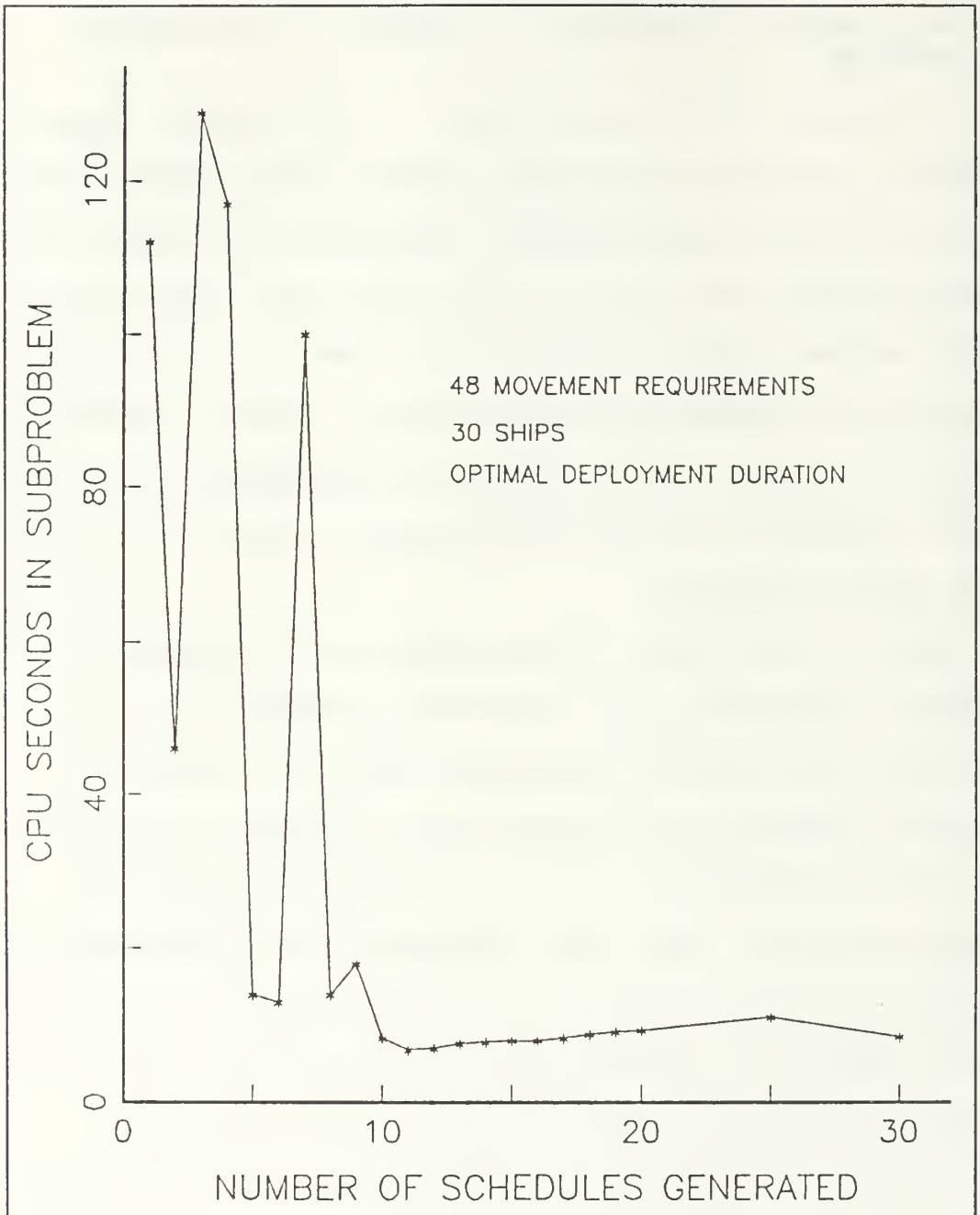

Figure 4-5

CPU time for subproblem versus number of schedules generated.

TABLE 4-6: AVERAGE NUMBER OF NODES GENERATED BY THE ALGORITHM FOR PROBLEM 2.

One column per cycle		Ten columns per cycle	
Iter	Average # nodes	Iter	Average # nodes
1-10	143	1-10	459
11-20	104	11-20	402
21-30	104	21-30	414
31-40	178	31-40	1482
41-50	149	41-45	315
51-60	139		
61-70	1323		
71-80	331		
81-90	297		
91-100	4999		
101-110	152		
111-120	13889		
121-130	290		
131-140	475		
141-150	19233		
151-160	8751		
161-169	483		

Figure 4-7 and 4-8 depict the trade-off between time spent in the master and subproblem for Problem 2. Figure 4-7 shows that by generating more columns per iteration one uses less CPU time for the subproblem. However, more columns per iteration also means more columns for the master problem which in turn leads to more CPU time for pricing out and other-housekeeping operations. In terms of total CPU time for the overall problem, Figure 4-8 shows that the 10 column strategy is better than the one column strategy when τ is less than or equal to the optimal duration (29 days). However, when τ is bigger than 29 days, the one column strategy is superior. This is due partly to the fact that when τ is larger than the optimal duration, the problem is more relaxed, i.e., it has a larger feasible region, and accurate dual information is not as critical as in the case when τ is smaller than the optimal duration.

Table 4-9 summarizes the computational results for all three problems. It is clear from this table that the ten column strategy dominates the one column strategy for all three problems. However, it would be premature to make any conclusive recommendations based on the results presented here.

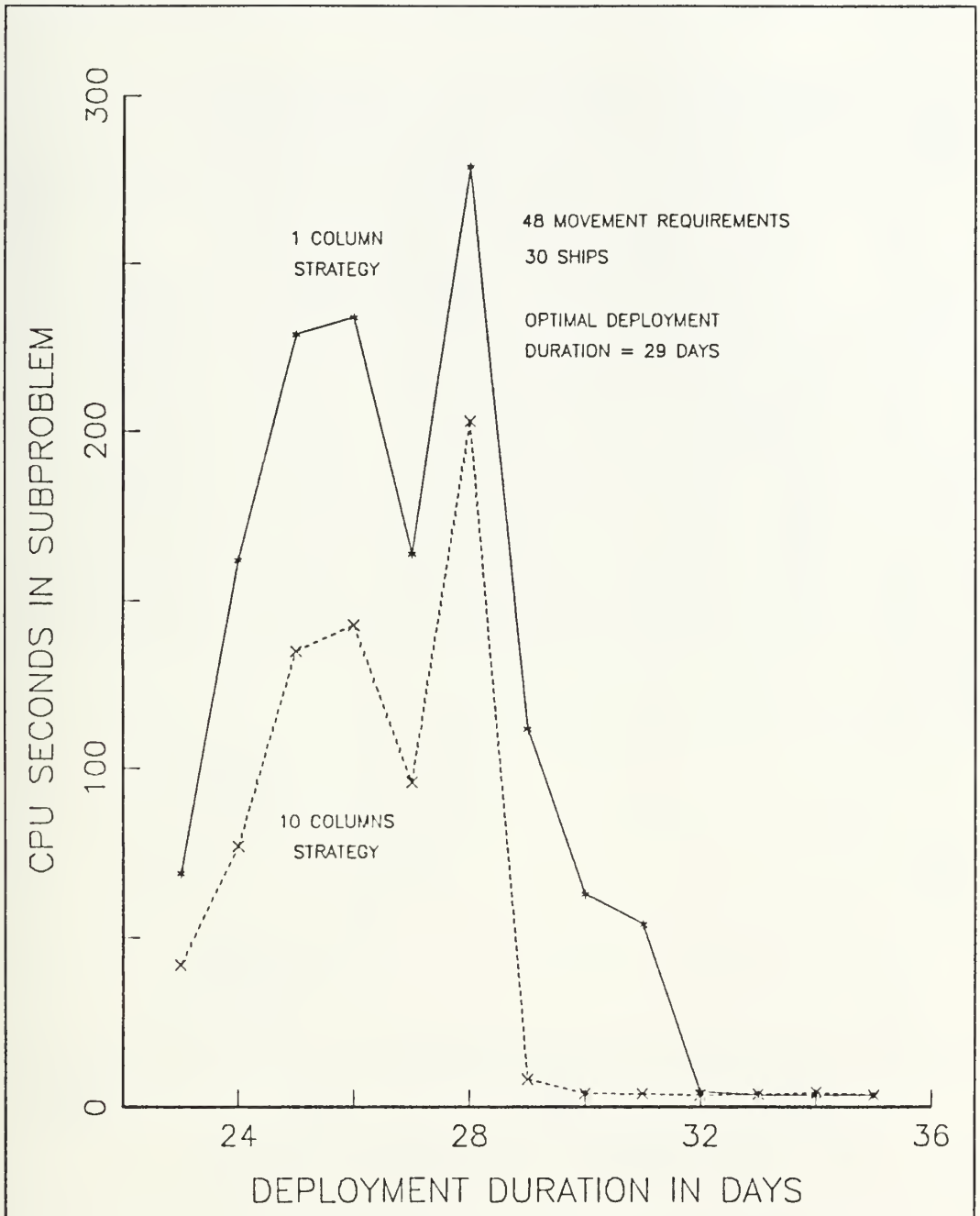

Figure 4-7

CPU time for subproblem versus τ

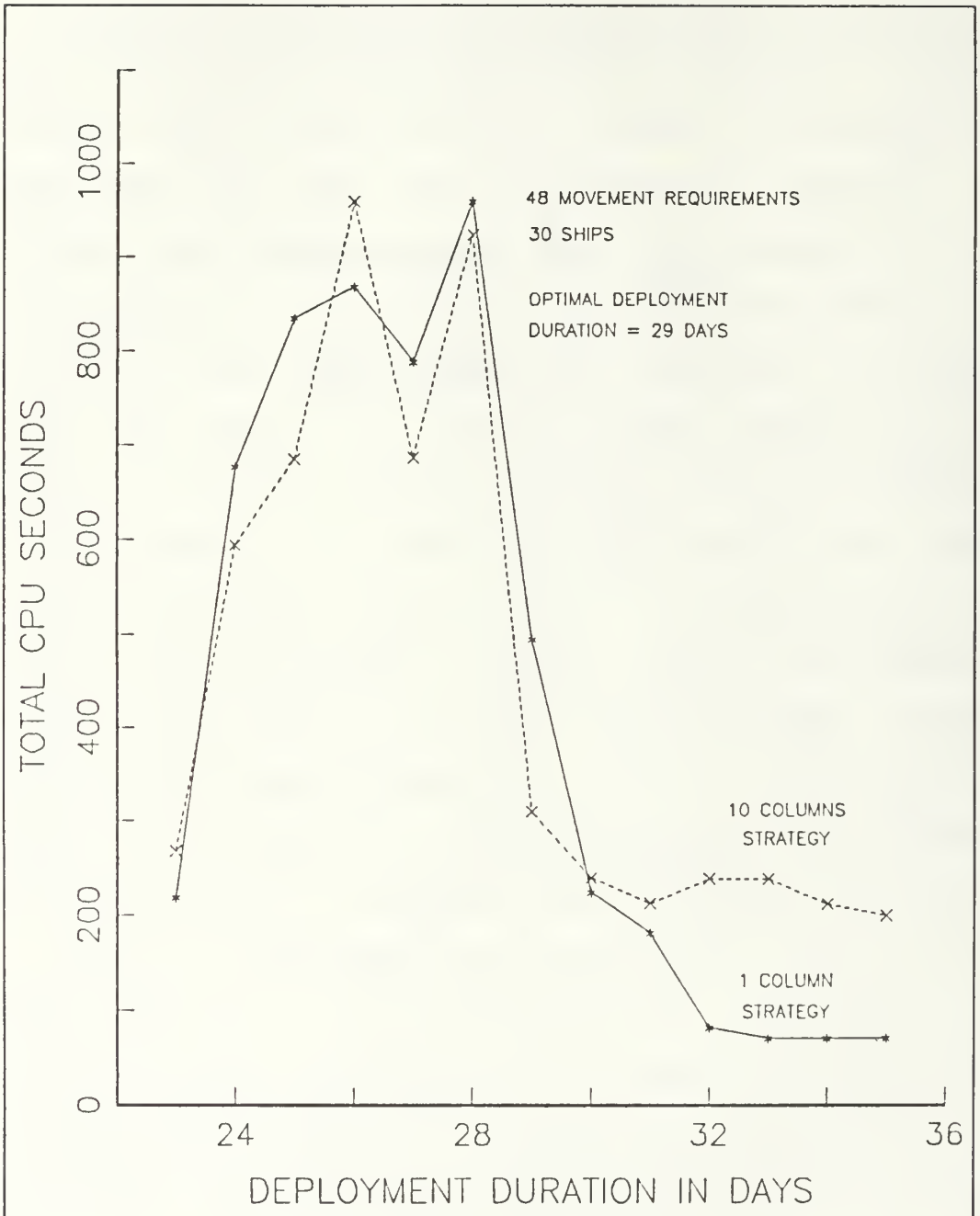

Figure 4-8

Total CPU time versus τ

TABLE 4-9 : COMPUTATIONAL RESULTS FOR THE THREE PROBLEMS.

	PROBLEM 1		PROBLEM 2		PROBLEM 3	
	1 col	10 col	1 col	10 col	1 col	10 col
τ	CPU time	CPU time	CPU time	CPU time	CPU time	CPU time
.8*OPT	307	94	69	42	413	406
.9*OPT	356	206	234	143	550	477
OPT	470	177	112	9	2048	447
1.1*OPT	18	5	5	4	4	5
1.2*OPT	4	2	3	4	10	5
Average	231	97	85	40	605	268

V. CONCLUSIONS AND FUTURE RESEARCH

An algorithm for generating schedules for the crisis deployment problem is described. This algorithm is similar to the branch and bound algorithm for integer programs or the depth-first search technique for traversing directed graphs. The computational results in Chapter 4 demonstrate that on medium size problems the algorithm, when integrated with the master problem algorithm by Lima [Ref. 10], is effective in solving the subproblem of the crisis deployment problem. However, a more extensive computational study is required before any conclusive recommendations can be made.

In addition to further computational studies, other possible areas for future research are listed below.

(1) Allow movement requirements to arrive at POE's within a time window.

(2) Allow movement requirements in partial shiploads. This would imply that a ship can pick up cargoes at two or more locations before delivering them to their discharging ports.

(3) Investigate other branching strategies for the algorithm. For example, one can branch to the closest movement requirement instead of the movement requirement with the most negative dual value.

(4) Investigate other combinatorial algorithms for the subproblem.

(5) Consider possible delays due to random events such as attacks on ships or ports.

APPENDIX

FORTRAN CODE FOR SUBPROBLEM

```
      SUBROUTINE SUBPR(U,XCOL,MDAY,M,N,NLOA,NPOE,NPOD,NSH,MR,TR,IT,A,
     &COMPAT,IB,XB,SHIP,XTIME,K,SPD,SEQ,LSEQ,CTSHIP,CTNODE,NPICK,NCOL)
C-----------------------------------------------------------------------
C THIS SUBROUTINE APPENDS NEW (GOOD) COLUMNS TO THE A MATRIX IN THE
C MASTERPROBLEM.
C
C KEY VARIABLES COMING FROM THE MASTER PROBLEM:
C
C  - U:        DUAL VARIABLES.
C  - MDAY:     MAXIMUM DURATION OF ANY SCHEDULE.
C  - A:        A MATRIX.
C  - COMPAT:   SHIP COMPATIBILITY WITH SHIPLOADS OF CARGO.
C  - IB:       INDEX SET FOR THE BASIS.
C  - XB:       COLUMNS IN THE BASIS.
C  - NPICK:    MINIMUM NUMBER OF SHIPLOADS OF CARGO TO BE PICKED BY
C              ANY SHIP.
C  - NCOL:     NUMBER OF SCHEDULES TO GENERATE FOR EVERY CALL TO THE
C              SUBPROBLEM.
C
C KEY VARIBLES USED IN THE SUBPROBLEM:
C
C  - VIND:     INDEX SET OF THE SORTED DUAL VARIABLES.
C  - LOAD:     SHIPLOAD NUMBER.
C  - PATH:     SEQUENCE OF SHIPLOAD NUMBERS FOR THE CURRENT SCHEDULE
C  - STACK:    SHIPLOAD NUMBERS IN THE STACK.
C  - CURLD:    CURRENT SHIPLOAD NUMBER.
C  - PRED:     PREDECESSOR OF SHIPLOAD NUMBER.
C  - TIME:     COMPLETION TIME FOR GENERATED SCHEDULES.
C  - LENGTH:   LENGT OF CURRENT PATH.
C
C
C-----------------------------------------------------------------------
      IMPLICIT DOUBLE PRECISION (A-H,O-Z), INTEGER (I-N)
      PARAMETER( MM =100, NN = 2000, KK = 2, JJ = 2000 )
      DIMENSION XCOL(MM),U(MM),UU(MM),V(MM),XB(MM),SA(MM),SPD(MM),IB(MM)
      REAL A(MM,NN)
      INTEGER VIND(MM),PRED(0:JJ),LOAD(0:JJ),TIME(0:JJ),STACK(0:JJ),TOP,
     & MR(100,KK),TR(15,15),IT(30,5),FROLD,TOLD,PATH(0:MM),CURLD,COUNT,
     & LENGTH,LASTND,SHIP,TT,MLNGTH,XTIME(NN),CTSHIP,CTBACK(0:MM),
     & CTSON(0:MM),SEQ(NN,MM),LSEQ(NN),CTNODE
      LOGICAL COMPAT(MM,MM)
      DOUBLE PRECISION MIN,MINRC
      COMMON /UNITS/ NIN,NOUT

C  INITIALIZE

      CTNODE = 0
      LIMIT = N
      CTSHIP = 0

311   DO 360 I = 1,MM
         SA(I) = 0.0D0
         UU(I) = 0.D0
         V(I) = 0.D0
         VIND(I) = 0

360   CONTINUE

      NNEG = NLOA
      MINRC = 0.0D0


C-----------------------------------------------------------------------
C  SORT THE DUAL VARIABLES
C-----------------------------------------------------------------------

      DO 8 I = 1,NLOA
8        UU(I) = U(I)

C  HEURISTIC:
C  ASSIGN LARGE NEGATIVE DUAL VALUE TO SHIPLOADS NOT ALREADY PICKED
C  BY OTHER SHIPS

      IF( N .GE. (2*NLOA + NSH + 1) ) THEN
         DO 201 I = 1,M
            DO 202 J = 1,M
               IF( (IB(J) .GT. (2*NLOA + NSH)) .AND.
     &             (XB(J) .GT. 0.5D0) ) THEN
                  SA(I) = SA(I) + A(I,IB(J))
               END IF
202         CONTINUE
                  IF( SA(I) .LT. 1.0D0 ) UU(I) = -2.0D0
201         CONTINUE
      END IF
```

```
C  REMOVE SHIPLOADS NOT COMPATIBLE WITH SHIP FROM FURTHER CONSIDERATION.

      DO 9 I = 1,NLOA
9     IF( .NOT. COMPAT(I,SHIP)  ) UU(I) = 99.0D0

      DO 20 I = 1,NLOA
            MIN = 0.1D-6
            COUNT = 0
            IND = 0
                DO 30 J = 1,NLOA
                      IF( UU(J) .LT. MIN ) THEN
                          MIN = UU(J)
                          IND = J
                          COUNT = 1
                      END IF
30                CONTINUE

              IF( COUNT .EQ. 0 ) THEN
                  NNEG = I - 1
                  GO TO 40
              END IF
            V(I) = MIN
            VIND(I) = IND

            UU(IND) = 99.0D0
20    CONTINUE

40    CONTINUE

*-------------------------------------------------------------------------*
*  DEPTH FIRST SEARCH
*-------------------------------------------------------------------------*

C  INITIALIZE

      DO 350 I =0,JJ - 1
         TIME(I) = 0
         STACK(I) = 0
         LOAD(I) = 0
         PRED(I) = 0
350   CONTINUE

      DO 380 I = 1,MM
         XCOL(I) = 0.D0
         PATH(I-1) = 0
         CTBACK(I-1) = 0
         CTSON(I-1) = 0
380   CONTINUE


      LENGTH = 0
      LASTND = 1
      TOP = 0
      CURLD = 0
      FROLD = 0
      TOLD = 0

C  CREATE ALL NODES OUT OF THE SOURCE AND PUT THEM IN STACK

      DO 50 I = NNEG,1,-1
          LOAD(NNEG - I + 2) = VIND(I)
          PRED(NNEG - I + 2) = 1
          STACK(NNEG - I + 1) = (NNEG - I + 2)
          TOP = TOP + 1
          LASTND = LASTND + 1
50    CONTINUE

C  MAIN LOOP TO SEARCH FOR FEASIBLE SCHEDULES

100   CURLD = STACK(TOP)

C  SPECIAL CASE WHEN ONE SHIP CAN PICK ALL SHIPLOADS OF CARGO

      IF( LENGTH .EQ. NNEG ) THEN
          RCOST = 0.0D0
          DO 91 I = 1,LENGTH
91        XCOL(LOAD(PATH(I))) = 1.0D0
          XCOL(NLOA + SHIP) = 1.0D0
          DO 111 I = 1,M
111       IF( XCOL(I) .EQ. 1.0D0 ) RCOST = RCOST + U(I)
          IF(RCOST .GT. -1.0D-4) RCOST = 0.0D0
```

```
          IF( (RCOST .LT. 0.0D0) ) THEN
               N = N+1
               DO 389 I = 1,M
389            A(I,N) = XCOL(I)
               DO 122 J=1,LENGTH
122            SEQ(N,J) = LOAD(PATH(J))
               LSEQ(N) = LENGTH
               RETURN
          END IF

        END IF

C  IF STACK EMPTY RETURN TO MASTERPROBLEM IF GOOD SCHEDULE FOUND.

        IF( CURLD .EQ. 0 ) THEN
          IF( MINRC .LT. 0.0D0 ) RETURN

C  IF NO GOOD SCHEDULES FOUND RUN SUBPROBLEM AGAIN FOR ANOTHER SHIP.

           CTSHIP = CTSHIP + 1
           SHIP = SHIP + 1
           IF(SHIP .EQ. NSH + 1) SHIP = 1
           IF( CTSHIP .NE. NSH ) GO TO 311
           RETURN
        END IF

      TOP = TOP - 1

      IF( PRED(CURLD) .EQ. 1) THEN
          LASTND = LASTND - LENGTH
          LENGTH = 0
      END IF


      IF( CTSON(LENGTH) .EQ. CTBACK(LENGTH) ) THEN
           DO 51 I = 1,LENGTH
              IF( PRED(CURLD) .EQ. PATH(I) ) THEN
                  LASTND = LASTND - LENGTH + I
                  LENGTH = I
                  GO TO 539
              END IF
51         CONTINUE
      END IF

539   PATH(LENGTH + 1) = CURLD

      FROLD = LOAD(PRED(CURLD))
      TOLD = LOAD(CURLD)
      TT = 0

C  CALCULATE COMPLETION TIME FOR CURRENT SCHEDULE.

      CALL TIMEFC(NPOE,NPOD,MR,TR,IT,FROLD,TOLD,SHIP,NSH,NLOA,TT,SPD)
      TIME(CURLD) = TIME(PRED(CURLD)) + TT

C  IF FEASIBLE INCLUDE SHIPLOAD NUMBER IN PATH.

      IF( TIME(CURLD) .LE. MDAY ) THEN

      CTBACK(LENGTH) = CTBACK(LENGTH) + 1
      LENGTH = LENGTH + 1
      CTSON(LENGTH) = 0

C  PUT INTO THE STACK ALL SHIPLOAD NUMBERS NOT ALREADY IN THE PATH.

          DO 60 I = NNEG,1,-1

              DO 70 J = 1,LENGTH
70                IF( VIND(I) .EQ. LOAD(PATH(J)) ) GO TO 60
                  LASTND = LASTND + 1
                  LOAD(LASTND) = VIND(I)
                  PRED(LASTND) = CURLD
                  TOP = TOP + 1
                  CTSON(LENGTH) = CTSON(LENGTH) + 1
60        CONTINUE

          DO 80 I = LASTND,(LASTND - CTSON(LENGTH) + 1),-1
              STACK(TOP) = I
              TOP = TOP - 1
80        CONTINUE
```

```
            TOP = TOP + CTSON(LENGTH)

          ELSE
              LASTND = LASTND - 1
              RCOST = 0.0D0

C   CHECK IF GOOD SCHEDULE FOUND.

              DO 90 I = 1,LENGTH
90            XCOL(LOAD(PATH(I))) = 1.0D0

              XCOL(NLOA + SHIP) = 1.0D0

              DO 110 I = 1,M
110           IF( XCOL(I) .EQ. 1.0D0 ) RCOST = RCOST + U(I)

              IF(RCOST .GT. -1.0D-4) RCOST = 0.0D0

             IF( RCOST .LT. 0.0D0 .AND. LENGTH .GT. NPICK ) THEN


                  IF ( CTBACK(LENGTH) .EQ. 0 ) THEN

                      N = N + 1
                      DO 388 I = 1,M
388                   A(I,N) = XCOL(I)

                      XTIME(N) = TIME(PRED(CURLD))

                      DO 522 J=1,LENGTH
522                   SEQ(N,J) = LOAD(PATH(J))

                      LSEQ(N) = LENGTH

                      IF( RCOST .LT. MINRC ) THEN
                          MINRC = RCOST
                          K = N
                      END IF

                  END IF

                  DO 400 I = 1,LENGTH
                      XCOL(LOAD(PATH(I))) = 0.0D0
                      XCOL(NLOA + SHIP) = 0.0D0
400               CONTINUE

              ELSE

                  DO 130 I = 1,LENGTH
130               XCOL(LOAD(PATH(I))) = 0.0D0
                  XCOL(NLOA + SHIP) = 0.0D0
              END IF

C   IF NO GOOD SCHEDULE FOUND BACKTRACK.

              CTBACK(LENGTH) = CTBACK(LENGTH) + 1
              GO TO 100

          END IF

          IF( N .GE. LIMIT+NCOL) RETURN
          CTBACK(LENGTH) = 0
          GO TO 100

          END
```

```
      SUBROUTINE TIMEFC(NPOE,NPOD,MR,TR,IT,FROLD,TOLD,SHIP,NSH,NLOA,TT,
     &SPD)
*-----------------------------------------------------------------------*
* THIS SUBROUTINE CALCULATES THE TIME REQUIRED TO PICK UP AND DELIVER   *
* A SHIPLOAD OF CARGO IN WHOLE DAYS.                                    *
*-----------------------------------------------------------------------*

      IMPLICIT DOUBLE PRECISION ( A-H,O-Z ), INTEGER ( I-N )
      PARAMETER( KK = 2, MM =100, NN = 2000 )
      DIMENSION SPD(MM)
      INTEGER MR(100,KK),TR(15,15),IT(30,5),TT,TOLD,SHIP,FROLD

      TT = 0

C   CALCULATING THE TRAVEL TIME

      IF(FROLD .EQ. 0) THEN
        TT = IDNINT((IT(SHIP,MR(TOLD,1)) + TR(MR(TOLD,1),MR(TOLD,2)))/
     &          (24. * SPD(SHIP)))
      ELSE

        TT = IDNINT((TR(MR(TOLD,1),MR(FROLD,2)) + TR(MR(TOLD,1),
     &          MR(TOLD,2)))/(24. * SPD(SHIP)))

      END IF

      RETURN
      END
```

LIST OF REFERENCES

1. Dantzig, G. B., and Wolfe, A., "The Decomposition Algorithm for Linear Programming," Econometrica, No. 29, pp. 767-778, 1961.

2. Rao, M. R., and Zionts. S., "Allocation of Transportation Units to Alternative Trips--A Column Generation Scheme with Out-of-Kilter Subproblems," Operation Research, Vol. 16, pp. 52-63, January 1968.

3. Appelgren, L. H., "A Column Generation Algorithm for a Ship Scheduling Problem," Transportation Science, No. 3, pp. 53-68, February 1969.

4. Brown, G. G., Graves, G. W., and Ronen, D., "Scheduling Ocean Transportation of Crude Oil," Management Science, Vol. 33, No. 3, pp. 335-346, March 1987.

5. Brown, G. G., Goodman, C. E., and Wood, R. K., Annual Scheduling of Atlantic Fleet Naval Combatants, Technical Report, Naval Postgraduate School, Monterey, CA, March 1985.

6. Ronen, D., "Cargo Ships Routing and Scheduling: Survey of Models and Problems," European Journal of Operations Research, Vol. 12, pp. 119-126, 1983.

7. Ronen, D., "Short-Term Scheduling of Vessels for Shipping Bulk or Semi-Bulk Commodities Originating in a Single Area," Operations Research, Vol. 34, No. 1, pp. 164-173, February 1986.

8. Collier, K. S., Deployment Planning: A Linear Programming Model With Variable Reduction, Masters Thesis, Naval Postgraduate School, Monterey, CA, September 1987.

9. Lally, M., Strategic Allocation of Sealift: A GAMS-Based Integer Programming Approach, Masters Thesis, Naval Postgraduate School, Monterey, CA, September 1987.

10. Lima, N. R., A Column Generation Technique for The Crisis Deployment Planning Problem, Masters Thesis, Naval Postgraduate School, Monterey, CA, September 1988.

11. Bodin, L., Golden, B., Assad, A., and Ball, M., "Routing and Scheduling of Vehicles and Crews: The State of the Art," Computer Operations Research, No. 10, pp. 63-211, 1983.

12. Aho, A. V., Hopcroft, J. E., and Ullman, J. D., Data Structures and Algorithms, pp. 53-56, Addison-Wesley Publishing Co., Reading, PA, 1983.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
2.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
3.	Lt. Cdr. Svein Buvik FO/SST/ORG 3 Oslo mil/Huseby 0016 Oslo 1 Norway	2
4.	Professor Siriphong Lawphongpanich, Code 55 Department of Operations Research Naval Postgraduate School Monterey, CA 93943	11
5.	Professor Richard E. Rosenthal, Code 55R1 Department of Operations Research Naval Postgraduate School Monterey, CA 93943	1
6.	Lt. Newton Rodrigues Lima Brazilian Naval Commission 4706 Wisconsin Ave. N. W. Washington, D.C. 20016	2